Movie Ticketing Bot

Rishi Dinesh

Introduction

Overview

Chatbots are intelligent programs that can understand human language, and cater to their users' needs by performing a diverse number of tasks. They are thus usually present as a layer on top of a service and act a gateway to the service. Today, most companies have chatbots to engage their users and serve customers by catering to their queries. Consumers spend lots of time using messaging applications (more than they spend on social media). Therefore, messaging applications are currently the most popular way companies deliver chatbot experiences to consumers. Hence, the goal of a chatbot is to interact with its users and deliver the company's services to them.

This project will be exploring the domain of the cinema industry to see how chatbots can be integrated into their system to help improve customer experience.

Purpose

The goal of this project is to develop a text-based chatbot that can be employed as a hep desk executive for a certain cinema provider (PVR cinemas, Chennai) and assist them in booking movie tickets while also catering to their other movie related queries. The chatbot will be built using IBM's Watson assistant. A sample UI that can act as a portal for its users to connect to PVR's services will also be built using Node-RED.

Literature Survey

Existing Problem

The main goal of a cinema provider's website would be to sell movie tickets online. As a digital platform , these companies face various challenges and are always looking to improve their digital presence. Some of the problems they face include increasing the online ticket sales, providing excellent customer service, and ultimately branding themselves as the leading innovative company.

Their websites usually have a million visitors each month that need help with getting answers to their questions, looking for movies recommendations, and booking tickets. These days people are searching for quick solutions and are not interested in waiting in line for a representative to become available. They want to get answers relevant to them and order tickets quickly and

easily from their sofa at home, their office or via their smartphone wherever they are. This is where chatbots come into the picture.Various approaches were taken to use chatbots to solve the above mentioned problems; a few of them are highlighted below.

[1] used a semi-automatic intelligent system called Chappie that was powered by AI/NLP to provide coherent messages from a business point of view. It could predict user's location and recommend movies based on various user inputs such as age, gender,etc. It also provided a hassle-free movie ticket booking service for its users.

Odeon's chatbot [2] on the other hand, developed by the social technology company Gruvi, required user to like the brand’s Facebook page and then either click “Message” or type “Odeon” into a chat search. After a greeting from the bot Users were asked for their location or what film they are interested in seeing. The bot then informed the customer of nearby cinemas or where, and what time, their selected film was showing. Once a decision had been made, the customer was sent a link to a booking page. This was developed in Europe for Odeon’s Cinemas.

[3] explored a ticketing chatbot service using serverless NLP technology. It details various NLP methods and serverless programming models and explains how they can be integrated with social media. It performed research where chatbots could act as customer service, and reported an F-score of 89.65% based on the conducted scenarios.

[4] developed a chatbot-based inquiry and ticket booking system for cinemas because they realized that there was no centralized database that contains the cinema and movie details, which caused the customers the need to obtain the info manually. Moreover, the existing ticket booking system is also not user-friendly due to the complex and non-standardized interface. It included to English and Bahasa Malaysia translation and was only focused on cinema in Malyaisa.

Many other approaches were taken to use chatbots to solve the problems of the existing system and these methodologies have been included in the Appendix section.

Proposed Solution

The solution to the above mentioned problems is a chatbot that can book movie tickets and answer users' movie related queries. For the sake of demonstration, this chatbot will be connected to a small database that has 5 movies with each movie having 2 to 3 different show times and each showtime having a different number of available seats. However, in a real life scenario, this chatbot would collect movie data directly from the cinema provider's website through API calls. The details about the sample database is given in the table below:

Movie Ticketing Bot using Watson Assistant and Node-RED

Master (Tamil)		Tenet (English)		Joker (English)		Dear Comrade (Telugu)		Darbar (Tamil)	
Showtime	# Seats	Showtime	# Seats	Showtime	# Seats	Showtime	# Seats	Showtime	# Seats
9:30AM	150	11:30AM	100	9:30AM	45	3:00PM	100	3:00PM	65
3:00PM	120	3:00PM	80	10:30PM	145	7:00PM	130	7:00PM	95
7:00PM	130	10:30PM	50	-	-	-	-	10:30PM	110

The chatbot will be designed to perform the following tasks:

- Provide different movie information
- Display various show times
- Answer inquiries about the availability of seats
- Respond to questions about the price of tickets
- Assist the user in booking movie tickets

The chatbot will be built using IBM's Watson assistant. Various intents and entities will be used in order to help the chatbot understand its users. Intents help the chatbot extract the goal the customer had in mind from his/her question/comment. In this project, 7 different intents will be used to capture the customer's various objectives:

- #Greetings : This intent will be used to identify when the customer greets the chatbot. This usually occurs at the beginning of a conversation and is used by the chatbot to provide an appropriate response depending on the nature of the greeting.
- #Movies : This intent will be used to identify when the user wants inquire about a particular movie or when he/she wants to book tickets for one.
- #Showtimes : This intent will be used to identify when the user wants to inquire about the various show times available at the cinema or even for a particular movie.
- #PriceInquiry : This intent will be used to identify when the user wants to inquire about the price of a ticket.
- #SeatInquiry : This intent will be used to identify when the user wants to inquire about the number of seats available.
- #CustomerNames : This intent will be used to identify customer names in the conversation. This will then be stored by the chatbot and printed in the booking confirmation message.
- #BookTickets : This intent will be used to identify when the user wants to book movie tickets. Once the assistant recognizes this intent, it will proceed to gather various customer details such as customer name, movie name, date of booking, preferred showtime, number of seats, the type of ticket (Elite or Economy) and the phone number. It uses this information and responds with the confirmation message.

Rishi Dinesh

In addition to intents, the chatbot will also make use of different entities in order to grasp the specifications of the user's question/comment and provide accurate and relevant responses. In this project, 6 different entities will be used to capture the different types of specific details in the user's intent:

- @Greetings : This entity will be used to identify the type of greeting given by the user. This could be "good morning" or "good evening" or even "good afternoon". This is used to help the chatbot deliver a more accurate greeting in response
- @Movies : This entity will be used to capture the exact movie that the user is trying to enquire about. This could one of the 5 movies mentioned in the table above.
- @Showtimes: This entity will be used to identify the exact showtime for which the user is trying to get more details. This could be one of the 5 different show times mentioned in the table above.
- @CustomerNames : This entity is works with the #CustomerNames intent to capture the user's name in the conversation and store it for future use.
- @PhoneNos : Similar to the one above, this entity will be used to identify the user's phone number and store it for future use.
- @PriceInquiry : This entity is used to identify the type of ticket (Elite or Economy) that the user is trying to enquire about.

Besides this, system entities such as @sys-date and @sys-number will be used to extract any date mentions and number mentions from the conversation. These intents and entities will be used to build dialog nodes that will allow the chatbot to fulfill the various tasks mentioned previously.

Each of these dialog nodes will have conditional responses based on the value of the entity recognized. If the customer wants to book tickets, the various customer details will be gathered using context slots and stored in context variables. Context slots will also be used to help the chatbot gather the information it needs before it can respond accurately to the user. This is the case of seat inquiries, where the number of seats available differ depending on the movie and the showtime.

The UI for the chatbot will be built and deployed using Node-RED. This frontend will be consist of a form where the user can submit his/her queries/comments and get back the chatbot's response. The Node-RED flow will consist of the following nodes in the given order:

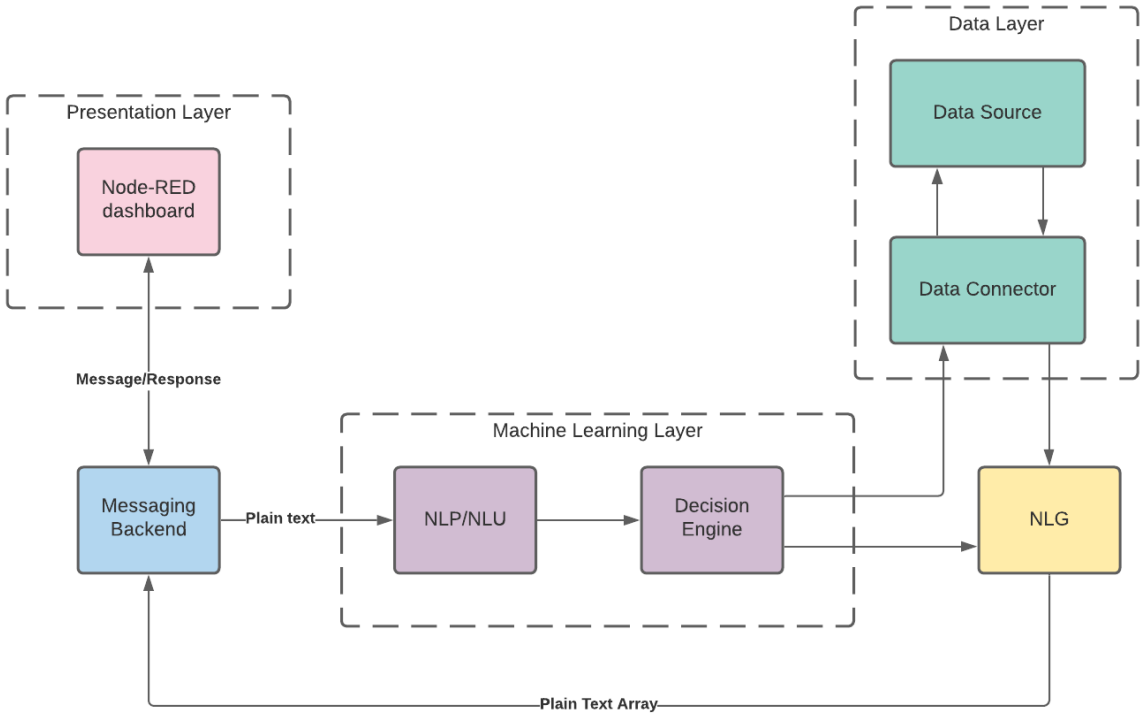
- A form node to take in the user's input.
- An input parsing function node that extracts and passes along the input as a message payload.
- The Watson conversation assistant node that connects the chatbot to the UI. This node takes in the user input and outputs the bot response.

- An output parsing function node to parse the output. The bot response can be a combination of text, image and options. This node processes this combination into a single message object and passes it along.
- An output formatting node that formats the output into a presentable format.
- Finally, two UI template nodes to display the user's input and the bot's formatted output.

The chatbot will be trained on numerous examples to help it accurately break down a given sentence into intents and entities. These examples will also be diverse and will be written from an end-user point of view in order to best capture all possible forms of a dialog.

Theoretical Analysis

Block Diagram



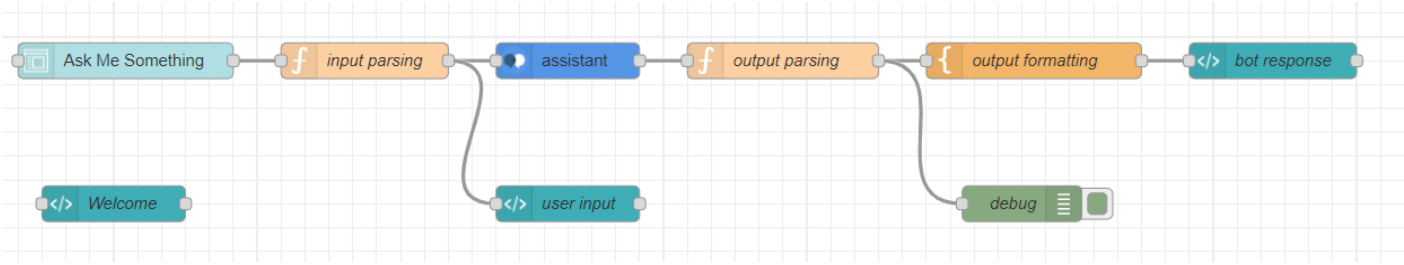
The above diagram shows the working of a general chatbot. The presentation layer is usually a Web UI or Social Media platform like Facebook but in this project, the Node-RED dashboard is used as the frontend. Messages from the user are passed to the messaging backend which in this case, is Watson assistant.

The assistant uses NLP (Natural Language Processing) and NLU (Natural Language Understanding) in order to extract the intents and entities from the message. It then passes this to the decision engine that uses this information to generate an appropriate response.

The decision engine is represented by the various dialog nodes used by Watson assistant. Each of these nodes access a data source to gather the required data to make build an appropriate response. In this project, the data source is a small sample of 5 movies with different show times that has been built-in to the assistant. However, in a real-life scenario, this source would be the cinema provider's website and the data connector would be the API that is used to query the data.

Once the bot has identified an appropriate response, it uses NLG (Natural Language Generation) to create a user-understandable and conversational output. This output is then passed back to the messaging backend which is in turn relayed to the frontend.

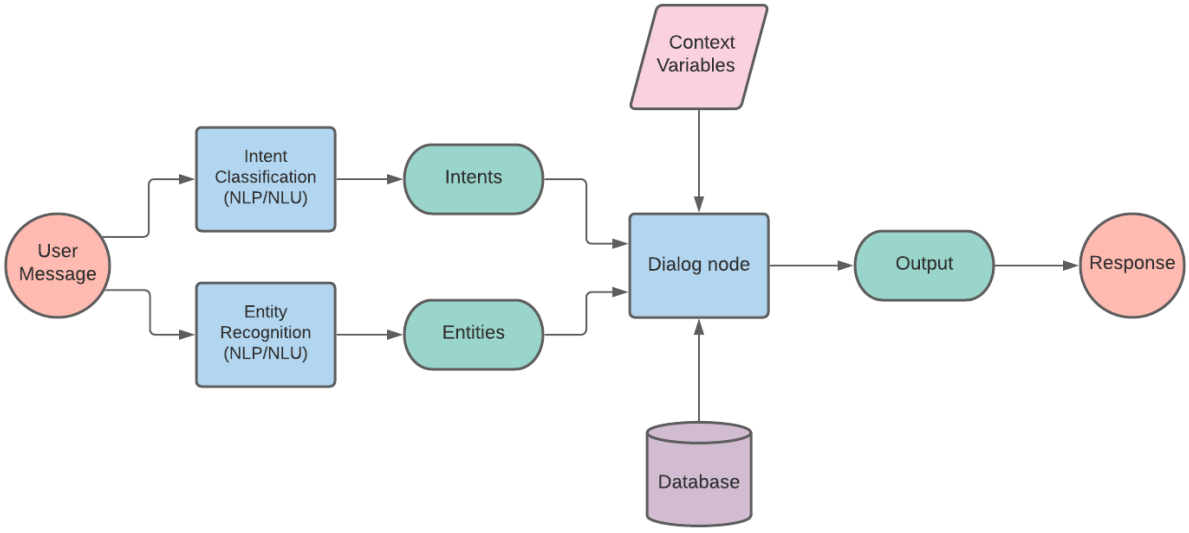
Software Design



The above diagram shows the Node-RED flow of the entire application. The user enters an input to the form labelled "Ask me Something". This input passed on to an input parsing function that transfers this input as the payload of a message object. These message object are what pass between nodes in the flow. They are plain JavaScript objects that can have any set of properties (like the payload).

The message is then sent to Watson assistant which in turn outputs the appropriate response (the working of the assistant is detailed in the next image). This output can be a combination of text, image and options. The output parsing function attaches the output to different attributes of the payload of another message object and also sets default values for missing attributes. It then passes on this message object to the output formatting function. This function extracts the payload from the message and adds some html around it.

This formatted output is then sent to a template node labelled "bot response" which essentially renders the html with the bot's output on a webpage. Another template node labelled "user input" also captures and displays the user's form input on the same webpage



The above diagram shows a simplified software architecture of Watson assistant. The user's message is first passed through an entity recognizer and intent classifier in order to extract the intents and entities of the message. This is done using various NLP and NLU techniques. In order for this step to produce accurate results, Watson needs to be pre-trained with a variety of examples for each entity and intent to help it better label the given message with the correct values.

These intents and entities are used to select the right dialog node to generate a response. This node interacts with the database and the context variables to build an appropriate output. The node gathers any required additional information with the help of context slots. Finally the output is sent back to the user as the bot's response.

Experimental Investigations

Various test cases were developed for each test scenario to check the correctness of entity recognition and intent classification and in turn to test if the chatbot could fulfill all the tasks mentioned previously. Manual testing was performed for each of the test cases, where all the possible inputs where executed manually without using any automated tool. Watson was able to accurately label all the intents and entities of each of the input sentences. This was largely due to the diverse number of examples that it was trained on prior to testing. Given in the table below is one test case from each test scenario:

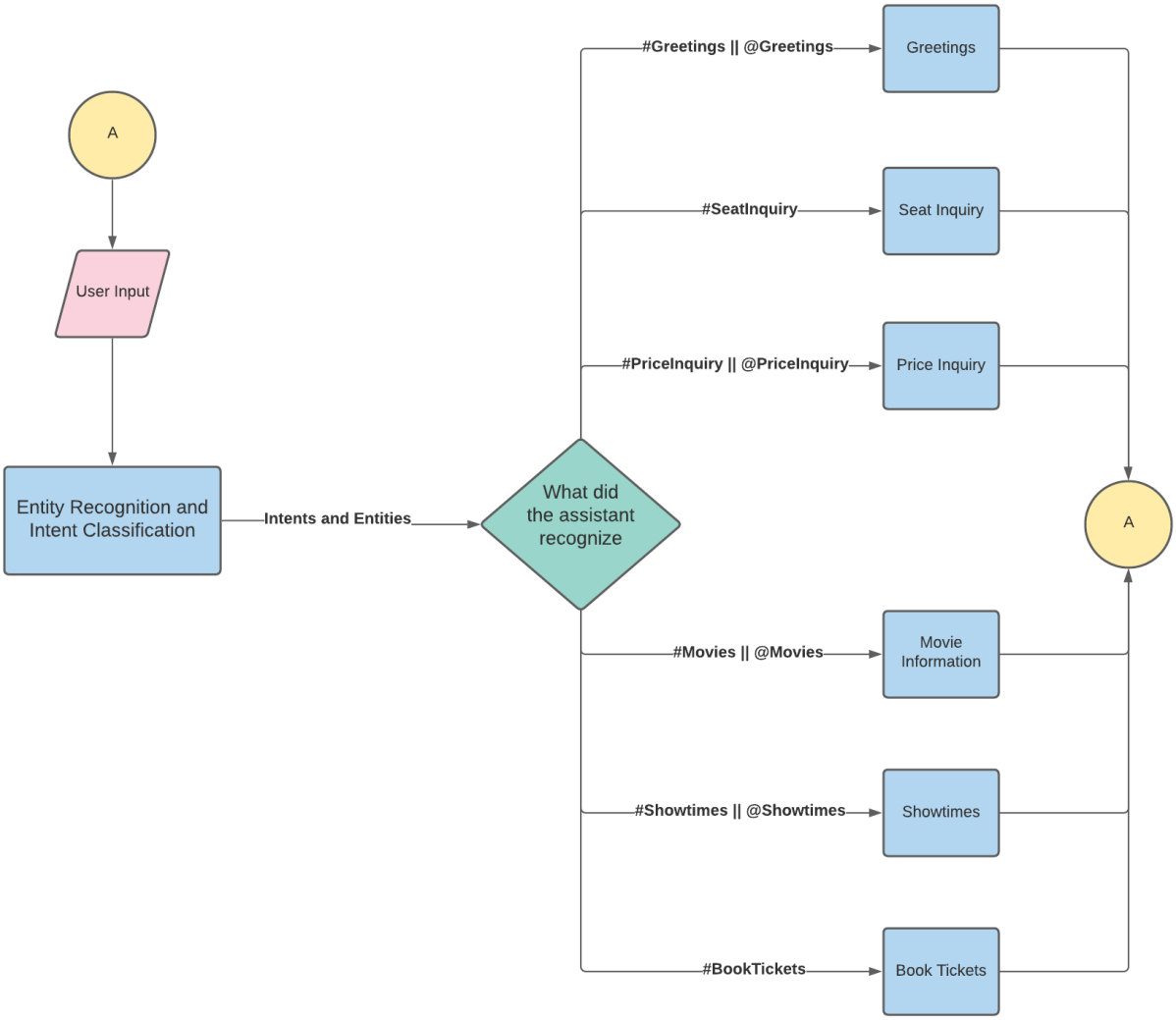
Movie Ticketing Bot using Watson Assistant and Node-RED

Test Case #	Test Scenario	Sentence Input (test case)	Expected Result	Actual Result	Pass/ Fail
1	Greetings	*"gud mrng"*	Intent : @Greetings:good morning Entity: #Greetings	Intent : @Greetings:good morning Entity: #Greetings	Pass
4	Show times	*"What are the timings for Master"*	Intent: #Showtimes Entity: @Movies:Master	Intent: #Showtimes Entity: @Movies:Master	Pass
5	Movie Information	*"tell me more about Darbar"*	Intent: #Movies Entity: @Movies:Darbar	Intent: #Movies Entity: @Movies:Darbar	Pass
6	Seat Inquiry	*"how many seats are available for Master at 9:30 AM"*	Intent: #SeatInquiry Entity: @Movies: Master @Showtimes: 9:30	Intent: #SeatInquiry Entity: @Movies: Master @Showtimes: 9:30	Pass
7	Price Inquiry	*"What is the cost of one Economy ticket"*	Intent: #PriceInquiry Entity: @PriceInquiry:Economy	Intent: #PriceInquiry Entity: @PriceInquiry:Economy	Pass
8	Book Tickets	*"I would like to 4 book tickets for Master at 9:30 AM under the name of John"*	Intent: #BookTickets Entity: @sys-number: 4 @Movies: Master @Showtimes: 9:30 @CustomerNames: John	Intent: #BookTickets Entity: @sys-number: 4 @Movies: Master @Showtimes: 9:30 @CustomerNames: John	Pass
9	Error handling	*"who is this"*	Irrelevant	Irrelevant	Pass

.

Rishi Dinesh

Flowchart



The flowchart given above is a high-level overview of how the user input is processed by the chatbot. The message is first broken down in to its entities and intents and based on the resultant output, one of the 6 dialog nodes are executed by the chatbot. Each of these nodes has their own conditional responses and context slots that allow the bot to deliver accurate responses based on the details found in the user input. Once the bot delivers a response, it goes back to the start of the flowchart, and waits for another user input to carry the conversation forward.

Result

The results of the experiments showed that the bot was able to identify the intents and entities accurately and also deliver the right response to the user's message. In case more information was required to generate a response, the bot was also able to use the context variables and slots correctly to gather this information before providing a relevant response. Using conditional responses, the bot was also able to provide multiple responses to the same input, based on different conditions such as the value of a context variable or an entity. In case of an unexpected input, the bot was also able to execute the correct exception handling dialog node and generate appropriate exception messages.

Advantages and Disadvantages

One of the biggest advantages of this chatbot is that it is available 24/7 for its customers and can alleviate some of the stress from the company's employees by promptly answering to every question presented to them, even if they are repeated. This ensures that customers can find solutions to their problems, be it day or night. They also offer the benefits of convenience and accessibility and improve customer satisfaction. Users can book tickets with a few messages with the chatbot and can also get information on various kinds of movie-related queries through a familiar textual interface. This can increase sales and also save money as the company can cut down on customer support costs.

However, unlike humans, chatbots lack emotions. They can be too mechanical in their responses and cannot adjust the conversational flow based on the customers emotions, unlike customer service executives. They can also be difficult to create and can only handle basic questions. They cannot handle complicated queries or answer out of the script questions and hence companies need to have human customer service employees that can manage these for them. Chatbots also require constant maintenance. Since the company services and database can change with time, all these changes need to be programmed into the chatbot so that it has the most up to date information.

Applications

Being a movie ticket booking chatbot, it can be employed to be a help desk assistant in various cinemas and theaters. It can be built into self-service kiosks at movie theaters where it can answer basic queries and assist the user in booking movie tickets. It can also be integrated into the company's website so that users visiting the website can address all their needs using an interactive textual interface from the comfort of their homes. Being a chatbot, it can also be integrated with messaging apps such as WhatsApp. This way, users can message the chatbot from anywhere through their mobile phones.

Conclusion

In this project, a chatbot application was developed using IBM's Watson assistant and deployed with a simple web UI using Node-RED. The purpose of this bot was to cater to its users' movie-related queries and assist them in booking movie tickets. The bot was able to carry out the following tasks successfully: provide different movie information, display various show times, answer inquiries about the availability of seats, respond to questions about the price of tickets and finally, assist the user in booking movie tickets. Manual testing was performed to test each of the above scenarios with a variety of test cases. The chatbot passed all the tests and was also able to effectively handle any unexpected user input and keep the conversation going.

Future Scope

The scope of the chatbot can be extended by adding various functionalities. It can be programmed to handle other aspects of a cinema experience, such as food, snacks and parking. The database can be extended to add more movies or the chatbot itself can be reconfigured to use APIs to take in data. The application can also be upgraded by adding a text to speech converter (and vice-versa) so that the chatbot can handle voice enabled inputs and outputs.

Bibliography

[1] Prof.Sumeet Pate, Mr.Akshay Bhagat, Miss.Ruchita Bhoir, Mr.Ayush Sharma "Human-to-Machine Conversation Modeling for Movie Ticket Booking using NLP", International Journal for Research in Engineering Application & Management (IJREAM), 2018
[2] https://celluloidjunkie.com/wire/gruvi-launches-first-european-cinema-chatbot-odeon-cinemas/
[3] E. Handoyo, M. Arfan, Y. A. A. Soetrisno, M. Somantri, A. Sofwan and E. W. Sinuraya, "Ticketing Chatbot Service using Serverless NLP Technology," 2018 5th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE), 2018, pp. 325-330, doi: 10.1109/ICITACEE.2018.8576921.
[4] Khiu, Z. H. (2020). Chatbot assisted inquiry and ticket booking system for cinema (Doctoral dissertation, UTAR).
[5] N. A. Godse, S. Deodhar, S. Raut and P. Jagdale, "Implementation of Chatbot for ITSM Application Using IBM Watson," *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, 2018, pp. 1-5, doi: 10.1109/ICCUBEA.2018.8697411.

Appendix

Souce Code

```
[
    {
        "id": "72256184.3bf3e",
        "type": "tab",
        "label": "Flow 1",
        "disabled": false,
        "info": ""
    },
    {
        "id": "f3444e3.7b0dcb",
        "type": "ui_tab",
        "name": "Movie Ticket Booking Bot",
        "icon": "dashboard",
        "disabled": false,
        "hidden": false
    },
    {
        "id": "e96a1470.c473c8",
        "type": "ui_base",
        "theme": {
            "name": "theme-custom",
            "lightTheme": {
                "default": "#0094CE",
                "baseColor": "#000000",
                "baseFont": "Times New Roman,Times,serif",
                "edited": true,
                "reset": false
            },
            "darkTheme": {
                "default": "#097479",
                "baseColor": "#097479",
                "baseFont": "-apple-system,BlinkMacSystemFont,Segoe
```

```
    UI,Roboto,Oxygen-Sans,Ubuntu,Cantarell,Helvetica Neue,sans-serif",
33              "edited": true,
34              "reset": false
35          },
36          "customTheme": {
37              "name": "Untitled Theme 1",
38              "default": "#4B7930",
39              "baseColor": "#0f62fe",
40              "baseFont": "-apple-system,BlinkMacSystemFont,Segoe
    UI,Roboto,Oxygen-Sans,Ubuntu,Cantarell,Helvetica Neue,sans-serif",
41              "reset": false
42          },
43          "themeState": {
44              "base-color": {
45                  "default": "#4B7930",
46                  "value": "#0f62fe",
47                  "edited": true
48              },
49              "page-titlebar-backgroundColor": {
50                  "value": "#000000",
51                  "edited": true
52              },
53              "page-backgroundColor": {
54                  "value": "#a7a7a7",
55                  "edited": true
56              },
57              "page-sidebar-backgroundColor": {
58                  "value": "#000000",
59                  "edited": true
60              },
61              "group-textColor": {
62                  "value": "#5380d5",
63                  "edited": true
64              },
65              "group-borderColor": {
66                  "value": "#000000",
67                  "edited": true
68              },
69              "group-backgroundColor": {
70                  "value": "#000000",
71                  "edited": true
72              },
73              "widget-textColor": {
74                  "value": "#eeeeee",
75                  "edited": false
76              },
77              "widget-backgroundColor": {
78                  "value": "#0f62fe",
79                  "edited": true
80              },
81              "widget-borderColor": {
82                  "value": "#000000",
83                  "edited": true
84              },
85              "base-font": {
86                  "value": "-apple-system,BlinkMacSystemFont,Segoe
    UI,Roboto,Oxygen-Sans,Ubuntu,Cantarell,Helvetica Neue,sans-serif"
87              }
88          },
89          "angularTheme": {
90              "primary": "indigo",
91              "accents": "blue",
92              "warn": "red",
93              "background": "grey"
94          }
```

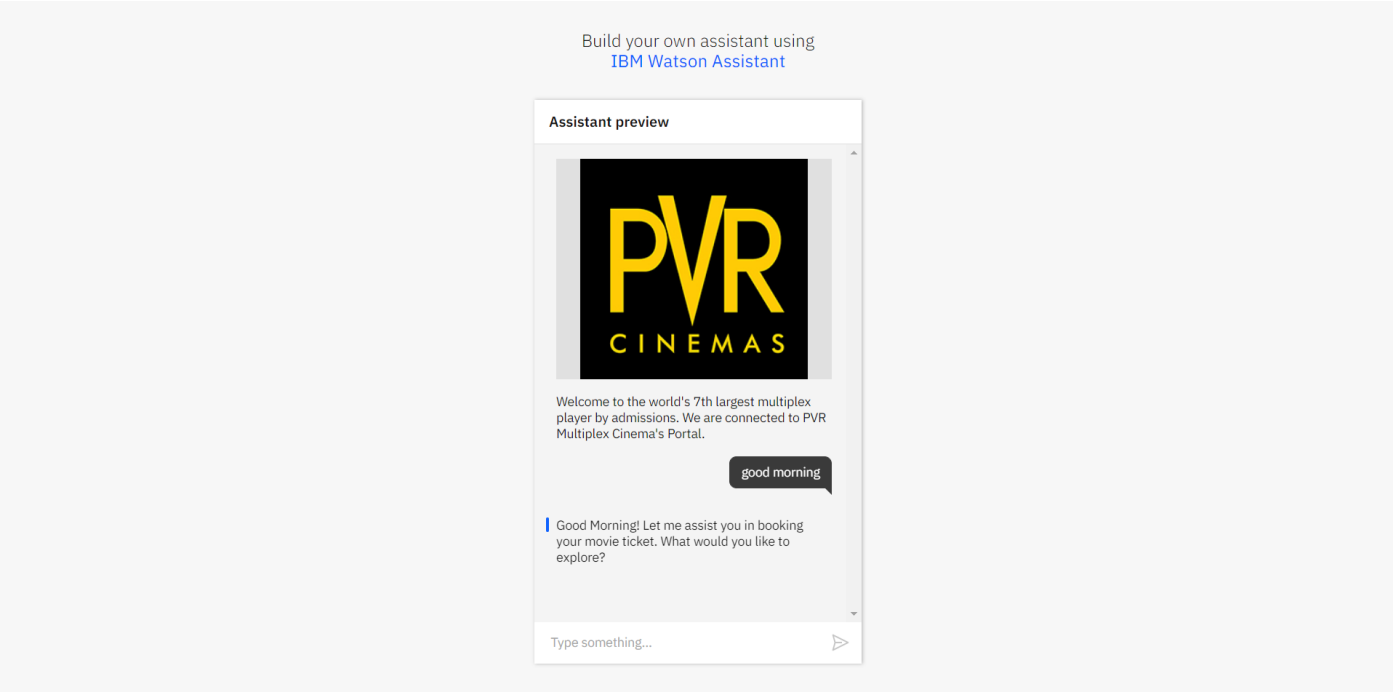
```
        },
        "site": {
            "name": "Node-RED Dashboard",
            "hideToolbar": "false",
            "allowSwipe": "false",
              "lockMenu": "false",
              "allowTempTheme": "true",
              "dateFormat": "DD/MM/YYYY",
              "sizes": {
                  "sx": 48,
                  "sy": 48,
                  "gx": 6,
                  "gy": 6,
                  "cx": 6,
                  "cy": 6,
                  "px": 0,
                  "py": 0
              }
          }
    },
    {
        "id": "678edef6.3d763",
        "type": "ui_group",
        "name": "",
        "tab": "f3444e3.7b0dcb",
        "order": 1,
        "disp": true,
        "width": "12",
        "collapse": false
    },
    {
        "id": "1425cb8f.4f4354",
        "type": "ui_form",
        "z": "72256184.3bf3e",
        "name": "",
        "label": "Ask Me Something",
        "group": "678edef6.3d763",
        "order": 2,
        "width": "12",
        "height": "12",
        "options": [
            {
                "label": "",
                "value": "input",
                "type": "text",
                "required": true,
                "rows": null
            }
        ],
        "formValue": {
            "input": ""
        },
        "payload": "",
        "submit": "submit",
        "cancel": "cancel",
        "topic": "topic",
        "topicType": "msg",
        "splitLayout": "",
        "x": 110,
        "y": 240,
        "wires": [
            [
                "ad5e8182.e0159"
            ]
```

```
        ]
    },
    {
        "id": "ad5e8182.e0159",
        "type": "function",
        "z": "72256184.3bf3e",
        "name": "input parsing",
        "func": "msg.payload = msg.payload.input;\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "libs": [],
        "x": 310,
        "y": 240,
        "wires": [
            [
                "7eb239b6.a01ba8",
                "b138e2d5.f2612"
            ]
        ]
    },
    {
        "id": "ac218362.4d7a1",
        "type": "function",
        "z": "72256184.3bf3e",
        "name": "output parsing",
        "func": "var outputs = msg.payload.output.generic\nvar default_img =
    \"https://media-cdn.tripadvisor.com/media/photo-s/0b/a1/e3/af/pvr-cinemas-4.jpg\"\nmsg.payload.text =
    \"\"\noutputs.forEach(output => {\n    var response = output.response_type\n    if( response ==
    \"text\"){\n        msg.payload.text += output.text\n        if(!msg.payload.url) msg.payload.url =
    default_img\n    }\n    else if(response == \"option\"){\n        msg.payload.title = output.title\n
    var options = []\n        output.options.forEach(opt => {\n
    options.push(opt.value.input.text)\n        })\n        msg.payload.options = options\n
    if(!msg.payload.url) msg.payload.url = default_img\n    }\n    else if(response == \"image\"){\n
    msg.payload.url = output.source\n    }\n})\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "libs": [],
        "x": 660,
        "y": 240,
        "wires": [
            [
                "b93b7277.bb63d",
                "51b3ae6.a754d5"
            ]
        ]
    },
    {
        "id": "7eb239b6.a01ba8",
        "type": "watson-conversation-v1",
        "z": "72256184.3bf3e",
        "name": "",
        "workspaceid": "602bce7d-e145-4caf-b478-397bb2b8212e",
        "multiuser": false,
        "context": true,
        "empty-payload": false,
        "service-endpoint":
    "https://api.us-south.assistant.watson.cloud.ibm.com/instances/c09d98d0-e888-4ca4-8233-c776a37292f2",
        "timeout": "",
        "optout-learning": false,
        "x": 480,
```

```
        "y": 240,
        "wires": [
            [
                "ac218362.4d7a1"
            ]
        ]
    },
    {
        "id": "94d8dc59.fb43",
        "type": "ui_template",
        "z": "72256184.3bf3e",
        "group": "678edef6.3d763",
        "name": "Welcome",
        "order": 1,
        "width": "12",
        "height": "14",
        "format": "<center><h1>PVR Assistant</h1></center>\n<center><h3><i>Welcome to the world's 7th largest multiplex player by admissions</i></h3></center>\n<img src=\"https://upload.wikimedia.org/wikipedia/commons/d/dc/Pvrcinemas_logo.jpg\">\n<h4><i>We are connected to PVR Multiplex Cinema's Portal. How can we help you?</i></h4>\n",
        "storeOutMessages": true,
        "fwdInMessages": true,
        "resendOnRefresh": true,
        "templateScope": "local",
        "x": 100,
        "y": 360,
        "wires": [
            []
        ]
    },
    {
        "id": "b138e2d5.f2612",
        "type": "ui_template",
        "z": "72256184.3bf3e",
        "group": "678edef6.3d763",
        "name": "user input",
        "order": 3,
        "width": "12",
        "height": "2",
        "format": "<i>YOUR INPUT</i> <h3>{{msg.payload}}</h3>",
        "storeOutMessages": true,
        "fwdInMessages": true,
        "resendOnRefresh": true,
        "templateScope": "local",
        "x": 480,
        "y": 360,
        "wires": [
            []
        ]
    },
    {
        "id": "b93b7277.bb63d",
        "type": "template",
        "z": "72256184.3bf3e",
        "name": "output formatting",
        "field": "payload",
        "fieldType": "msg",
        "format": "handlebars",
        "syntax": "mustache",
        "template": "{{payload.text}}\n<br>\n{{payload.title}}\n<br>\n{{#payload.options}}\n<ul>\n<li>{{.}}</li>\n</ul>\n{{/payload.options}}\n<br>\n<img src= {{payload.url}} width=\"600\" height=\"600\">",
        "output": "str",
        "x": 870,
```

```
        "y": 240,
        "wires": [
            [
                "4519a42a.8a7e1c"
            ]
        ]
    },
    {
        "id": "4519a42a.8a7e1c",
        "type": "ui_template",
        "z": "72256184.3bf3e",
        "group": "678edef6.3d763",
        "name": "bot response",
        "order": 4,
        "width": "12",
        "height": "15",
        "format": "<i>RESPONSE</i><br>\n<div ng-bind-html=\"msg.payload\"></div>",
        "storeOutMessages": true,
        "fwdInMessages": true,
        "resendOnRefresh": true,
        "templateScope": "local",
        "x": 1070,
        "y": 240,
        "wires": [
            []
        ]
    },
    {
        "id": "51b3ae6.a754d5",
        "type": "debug",
        "z": "72256184.3bf3e",
        "name": "debug",
        "active": true,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "payload",
        "targetType": "msg",
        "statusVal": "",
        "statusType": "auto",
        "x": 860,
        "y": 360,
        "wires": []
    }
]
```

Output Screenshots

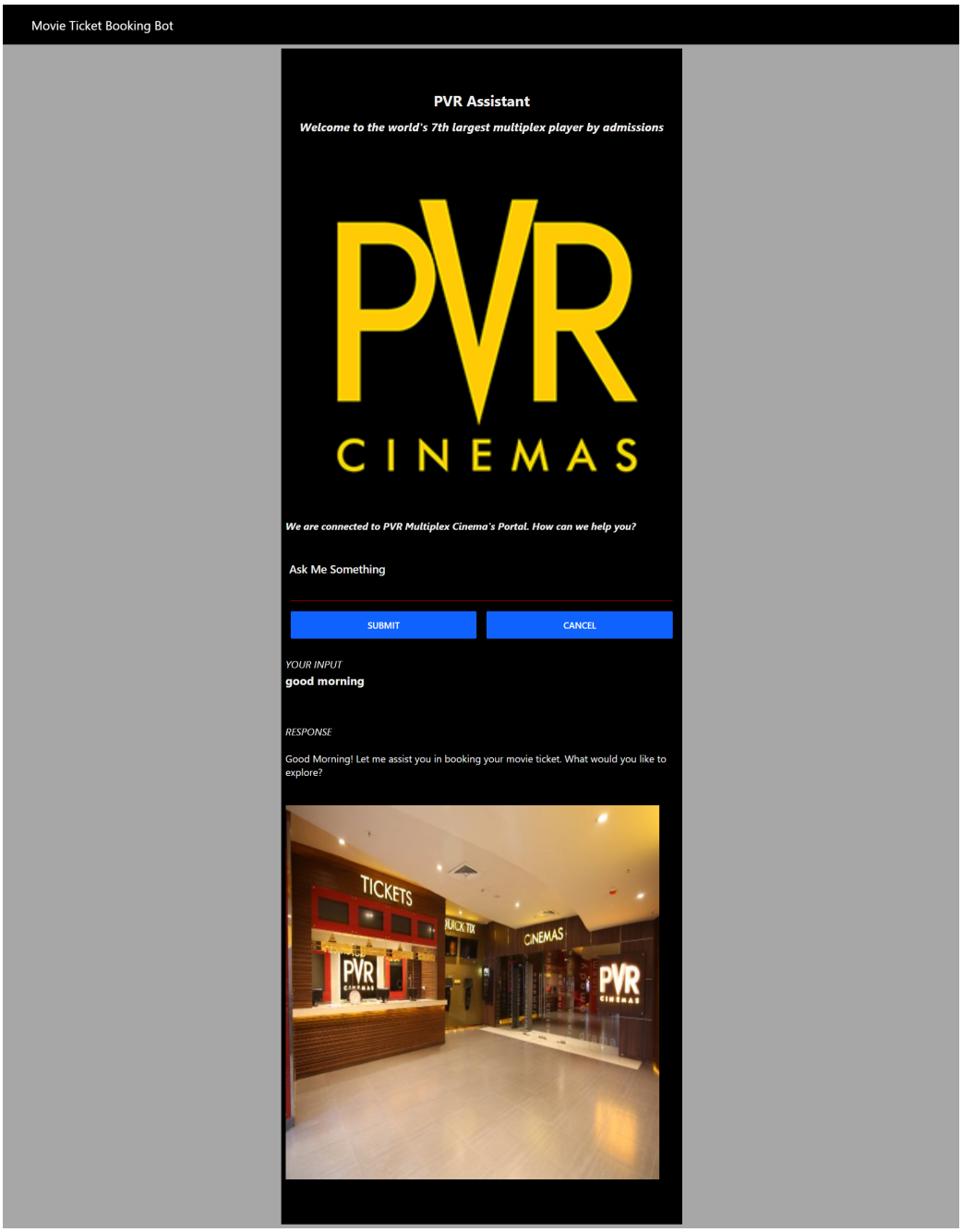


The above image shows the preview link of the chatbot built using IBM's Watson Assistant. It has a conversational interface where the user can type in a message and the bot will reply with the appropriate response.

Watson Assistant Preview Link

The image given below is a screen capture of the UI built using Node-RED. It also contains a form where the user can enter a message. The UI dashboard is connected to Watson assistant that takes in this message and produces the response as the output. This output can be a combination of text, image or options. A template node and an output parsing function is used to pretty print the output as the bot's response.

Movie Ticketing Bot using Watson Assistant and Node-RED



Rishi Dinesh